Universidade Federal de Minas Gerais

Escola de Engenharia

Disciplina: Projeto de Sistemas Embutidos

Relatório do Projeto: Solar Tracker

Autores:
Igor Cleto
Matheus Galbiatti

Belo Horizonte
10 de julho de 2025

Contents

1 Introdução

Este documento detalha o projeto e desenvolvimento de um protótipo de rastreador solar de eixo único (azimutal), concebido no âmbito da disciplina de Projeto de Sistemas Embutidos. O objetivo principal é criar uma solução de baixo custo e autossustentável para otimizar a captação de energia solar, abordando a limitação de eficiência dos painéis solares fixos. O projeto foi planejado utilizando a metodologia do Project Model Canvas para estruturar as ideias e garantir o alinhamento entre os objetivos, requisitos e restrições.

2 Project Model Canvas

O Project Model Canvas é uma ferramenta de gerenciamento visual utilizada para descrever, projetar e analisar modelos de projetos de forma concisa e integrada. Ele é composto por 13 blocos que cobrem as áreas fundamentais de um projeto, desde suas justificativas e objetivos até os custos e cronograma. A seguir, cada um dos blocos é detalhado conforme definido para o projeto Solar Tracker.

2.1 Justificativas

Painéis solares fixos apresentam uma limitação intrínseca na captação de energia, pois não acompanham o movimento aparente do sol, resultando em menor eficiência ao longo do dia.

2.2 Produto

Sistema de otimização para a captação de energia solar, que opera por meio do acompanhamento do azimute solar, sendo complementado pela documentação técnica.

2.3 Objetivo SMART

Desenvolver um protótipo funcional de um sistema de rastreamento solar de eixo único (azimutal) para aumentar a captação de energia de um painel solar de maneira autossustentável.

2.4 Requisitos

- **Rastreamento Solar Autônomo:** Detectar e seguir autonomamente a luz solar.
- **Movimentação:** Em um eixo (azimutal).
- **Simplicidade:** Componentes de hardware e software simples, com uma interface de usuário (IU) intuitiva.
- **Desempenho Operacional:** Operação confiável e ágil.
- **Restrição Orçamentária:** Desenvolvimento e operação dentro de um orçamento limitado.

2.5 Benefícios Futuros

O produto aprimora a eficiência energética e otimiza a utilização do espaço, enquanto o monitoramento remoto e uma solução de rastreamento solar de baixo custo complementam seus benefícios. O resultado é a otimização da eficiência, atendendo à necessidade de soluções energéticas práticas e eficazes.

2.6 Stakeholders e Fatores Externos

O projeto é influenciado por fornecedores de componentes eletrônicos e mecânicos (disponibilidade e preço), o ambiente climático (condições para testes) e potenciais usuários/avaliadores de protótipos na área de energia solar.

2.7 Equipe

- Igor Cleto: Validação do projeto.
- Matheus Galbiatti: Desenvolvimento do projeto.

2.8 Premissas

- Disponibilidade contínua dos componentes críticos no mercado.
- Condições de insolação solar consistentes e adequadas durante a fase de testes.
- Viabilidade de um design de sistema que assegure um consumo energético inferior ao ganho obtido pela captação solar.

2.9 Riscos

- Indisponibilidade de componentes críticos.
- Potenciais atrasos no cronograma estabelecido de 8 semanas.
- Condições climáticas desfavoráveis que prejudiquem os testes planejados.
- Consumo energético do sistema que supere os benefícios da captação solar.

2.10 Grupo de Entregas

1. Design e planejamento detalhado.
2. Desenvolvimento do protótipo de hardware e software.
3. Testes e refinamentos.
4. Protótipo funcional e documentação final.

2.11 Linha do Tempo

- **Semanas 1 e 2:** Design e planejamento.
- **Semanas 3 a 5:** Desenvolvimento do protótipo.
- **Semanas 6 e 7:** Testes e refinamentos.
- **Semana 8:** Finalização.

2.12 Custos

O projeto conta com um orçamento fixo de R$300,00, destinado exclusivamente aos componentes eletrônicos e mecânicos necessários para o protótipo.

2.13 Restrições

- Testes do protótipo excluem condições climáticas extremas.
- Os membros da equipe vão dedicar 5 horas por semana para o desenvolvimento e acompanhamento do projeto.

3 Hardware e Componentes

4 Backend

5 Frontend com Streamlit

O frontend do projeto é um dashboard interativo desenvolvido integralmente em Python com a biblioteca Streamlit. A escolha desta tecnologia foi estratégica, visando acelerar o ciclo de desenvolvimento ao eliminar a necessidade de linguagens de frontend tradicionais (HTML, CSS, JavaScript) e facilitar a integração direta com a lógica de controle e análise de dados do projeto.

5.1 Arquitetura e Fluxo de Dados

A aplicação Streamlit atua como o centro de controle e monitoramento do Solar Tracker. Sua arquitetura é baseada em um fluxo de dados em tempo real, sustentado por quatro tecnologias principais.

- **Streamlit:** É o framework que renderiza a interface web. Para garantir que a interface reflita os dados mais recentes, a biblioteca `streamlit-autorefresh` é utilizada para recarregar a página a cada 500 milissegundos.
- **Paho-MQTT:** A comunicação com o hardware (ESP32) é desacoplada e realizada através do protocolo MQTT. A aplicação se conecta a um broker EMQX Cloud e opera de duas formas:
  - **Subscrição:** Inscreve-se no tópico `esp32/angulo` para receber, de forma assíncrona, as atualizações de estado do painel. O payload esperado é uma string no formato `"angulo;azimute;manual;rele"`.
  - **Publicação:** Publica mensagens no tópico `esp32/comando` para enviar instruções ao microcontrolador, como a mudança de modo de operação ou a definição de um ângulo manual.
- **Pandas:** É utilizado para a gestão dos dados históricos. Cada mensagem recebida via MQTT é processada e salva como uma nova linha em um arquivo CSV (`dados_historicos.csv`), que armazena o timestamp, ângulo, azimute, modo manual e estado do relé. Isso garante a persistência dos dados.

5.2 Gestão de Estado e Atualização da Interface

A gestão de estado é crucial em aplicações Streamlit. O objeto `st.session_state` é utilizado para manter a persistência de dados e conexões durante a sessão do usuário:

- `st.session_state.mqtt_client`: Armazena a instância do cliente MQTT, garantindo que a conexão com o broker seja estabelecida apenas uma vez.
- `st.session_state.dados_atuais`: Um dicionário que guarda os dados mais recentes lidos do arquivo CSV, permitindo que sejam exibidos na interface.
- `st.session_state.historico`: Mantém o DataFrame do Pandas com todos os dados históricos.
- `st.session_state.operating_mode`: Armazena o modo de operação selecionado pelo usuário (Manual ou Automático).

A comunicação MQTT é tratada de forma assíncrona com `client.loop_start()`, que cria um thread em segundo plano. A função de callback `on_message` é responsável por receber os dados, processá-los e anexá-los ao arquivo CSV. A interface do Streamlit, por sua vez, é atualizada através do mecanismo de auto-refresh, que aciona uma função para recarregar os dados do CSV e redesenhar os componentes da tela com as informações mais recentes.

5.3 Detalhes da Interface de Usuário

A interface foi projetada para ser clara e funcional, dividida em um painel de controle e um dashboard principal.

5.3.1 Painel de Controle (Sidebar)

A barra lateral (`st.sidebar`) agrupa os controles interativos:

- **Modo de Operação:** Dois botões distintos (`st.button`), "Ativar Modo Automático" e "Ativar Modo Manual", permitem ao usuário alternar entre os modos. O estado do botão é desativado se o modo correspondente já estiver ativo, fornecendo um feedback visual claro. Ao clicar, um comando ('A' para automático, 'M' para manual) é publicado no tópico MQTT.
- **Controle Manual:** Visível apenas no modo "Manual", este controle utiliza um `st.slider` para uma seleção de ângulo intuitiva (0 a 180 graus). O comando só é enviado ao clicar no botão "Enviar Ângulo", evitando o envio contínuo de mensagens MQTT.

5.3.2 Dashboard Principal

A área principal foca na visualização dos dados em tempo real e histórico:

- **Status Atual:** Quatro métricas principais são exibidas usando `st.columns` para organização.
  - **Ângulo e Azimute:** Componentes `st.metric` mostram os valores numéricos atuais.
  - **Modo e Relé:** Para o modo de operação e o estado do relé, foi utilizado HTML personalizado dentro de `st.markdown` para criar indicadores visuais (LEDs) que mudam de cor (verde para ativo/ligado, vermelho para inativo/desligado), oferecendo um feedback rápido e intuitivo.
- **Gráfico Histórico:** O `st.line_chart` renderiza o histórico de ângulos e azimutes, utilizando o timestamp como índice para o eixo X.
- **Tabela de Dados:** O uso do `st.expander` permite que o usuário visualize a tabela completa de dados históricos (`st.dataframe`), mantendo a interface principal limpa e organizada.

As figuras 1 e 2 ilustram a aparência do dashboard em seus dois modos de operação principais.

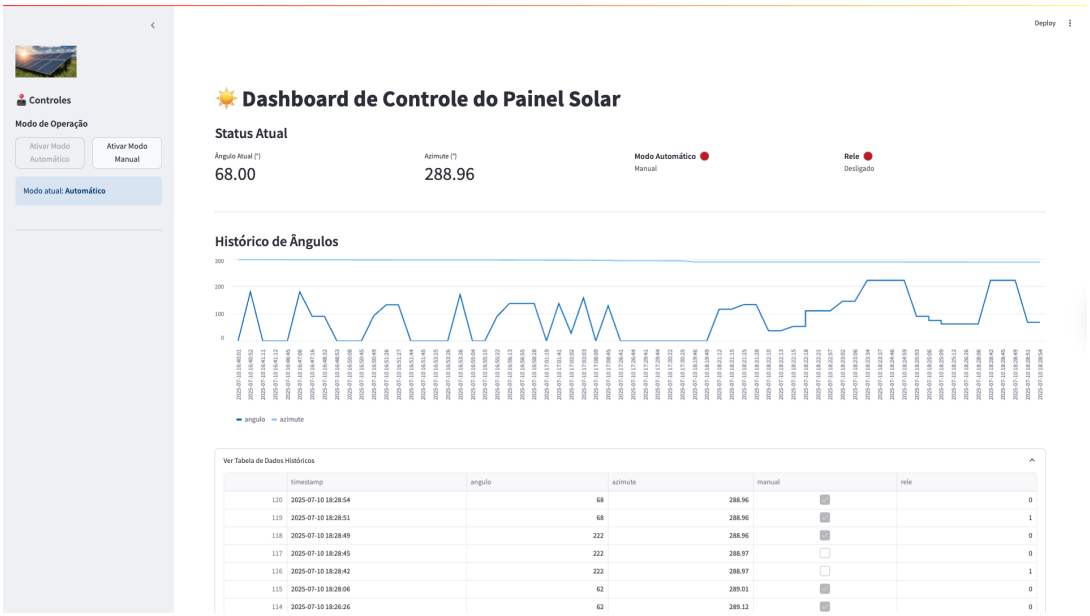

Figure 1: Interface do dashboard no modo automático.

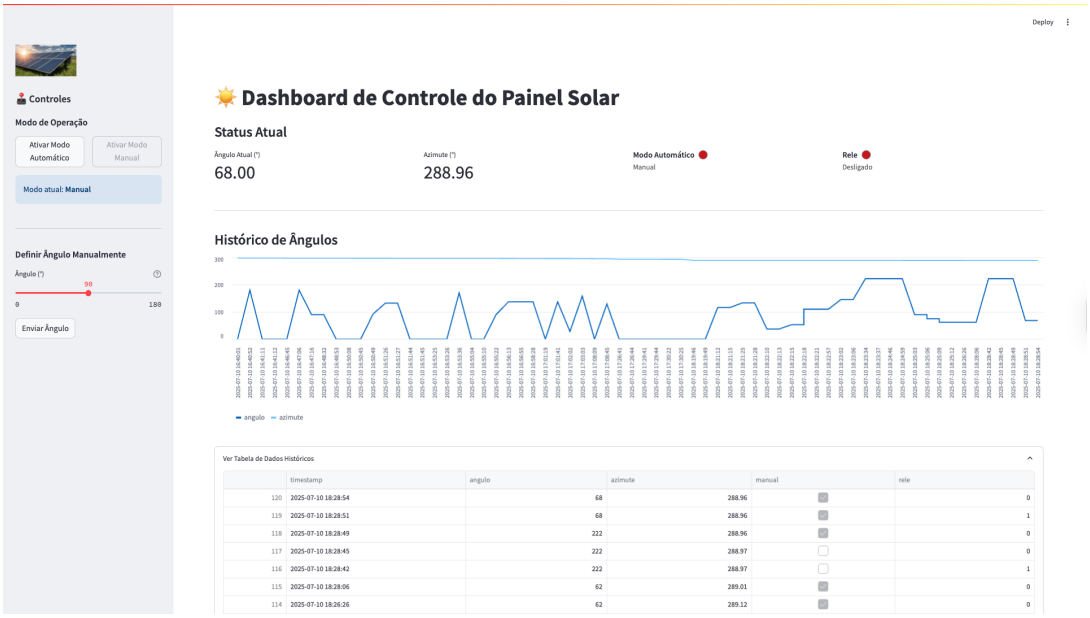

Figure 2: Interface do dashboard no modo manual, com o controle de ângulo visível na barra lateral.

6 Conclusão